C8. Auto-encoders & generative adversarial networks

Advanced Machine Learning (MLA)

M2 Engineering of Intelligent Systems

& Advanced Systems and Robotics

2022-2023

Plan

1. Unsupervised learning
   - Foundations, architectures, & applications
2. Auto-encoders (AE)
   - Definition and principle
   - Learning, regularization, and invariances
3. Generative adversarial networks (GAN)
   - Definition
   - Learning, alternate optimization, and issues
4. Conditional auto-encoders

Holy books

Books & papers

Goodfellow, I., Bengio, Y. & Courville, A.
Deep Learning. (MIT Press, 2016).

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In International Conference on Neural Information Processing Systems (NIPS), page 153–160.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786) :504–507.

Ranzato, M. A., l. Boureau, Y., and Cun, Y. L. (2007). Sparse feature learning for deep belief networks. In International Conference on Neural Information Processing Systems (NIPS), page 1185– 1192.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. In *International Conference on Neural Information Processing Systems (NIPS)*, page 2672–2680.

More readings on the web

Some blogs on AE & GAN

CVPR'18 tutorial on GAN

Blog de Jonathan Hui

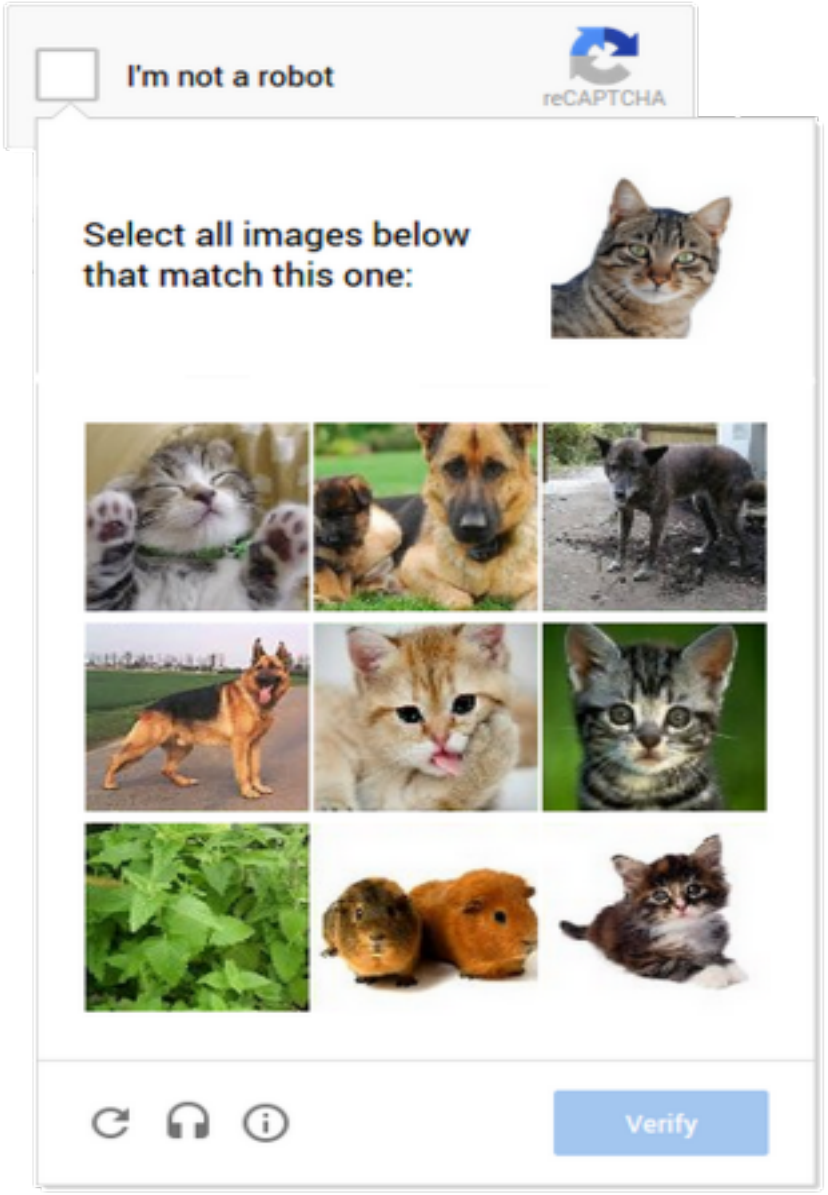
Le GAN Lab : https://poloclub.github.io/ganlab/

SCIENCES SORBONNE UNIVERSITÉ

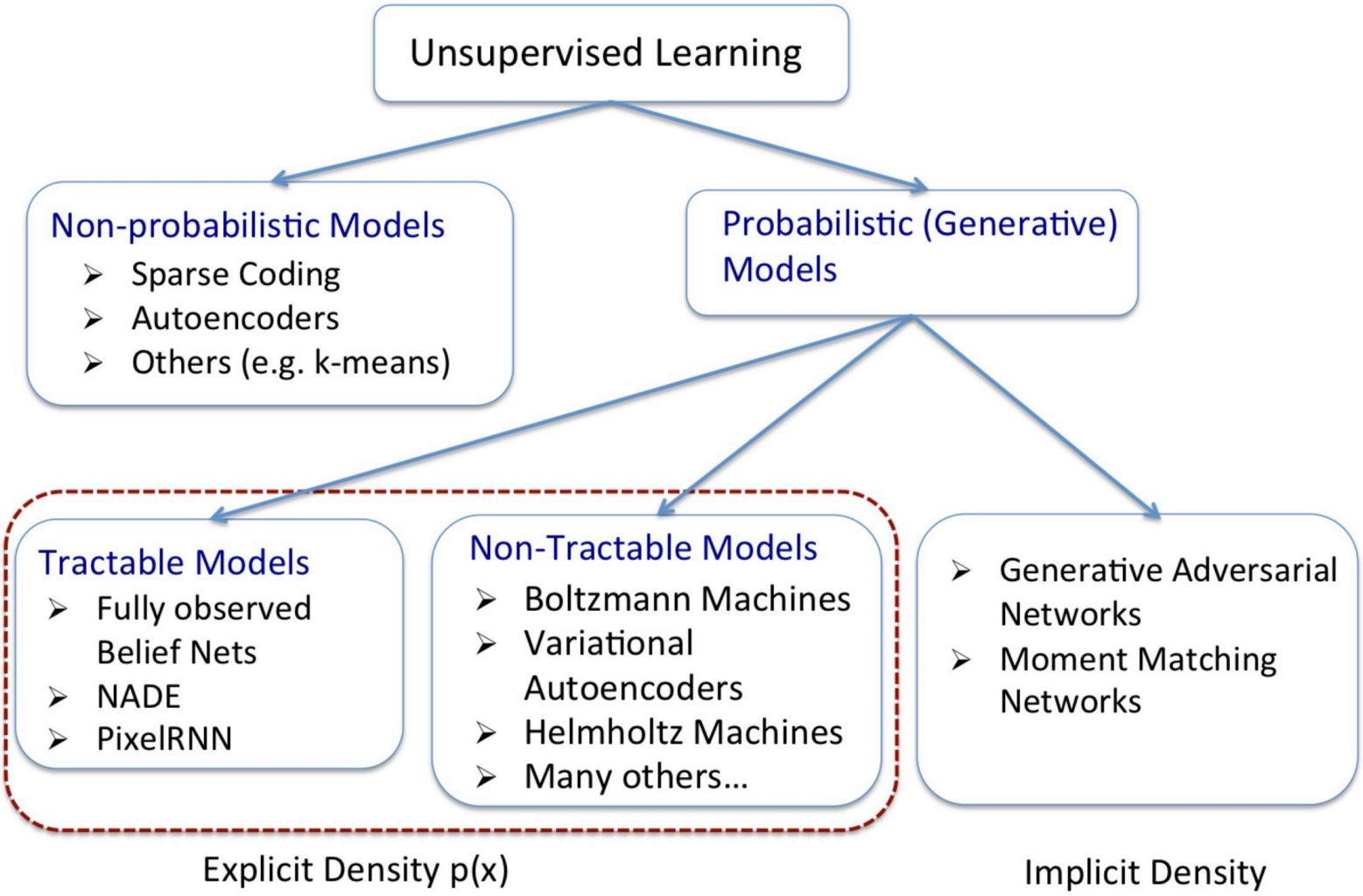
1. Unsupervised learning

Foundation

- Preliminary
  - Supervised learning is guided by the task
  - Requires ground truth input/output data to learn from
- Limitations
  - Deep learning is particularly efficient with large amount of data
  - But the human production of these data is extremely costly (in time and in resources)
- Idea
  - Can we learn something from the raw data?
  - Towards semi-supervised, slightly supervised, and unsupervised learning



Architectures



Sparse coding

Sparse coding has been originally developed for coding and data compression *[Olshausen and Field, 1996]*

Sparse coding is mathematically defined as a linear combination of bases Wk, such as

$$\mathbf{x}_n \approx \mathbf{v}_n = \sum_{k=1}^{K} a_{n,k} W_k$$

In sparse coding, the activation $a_{n,k}$ are encouraged to be sparse, i.e., with a maximum of nearly zero values

Sparse coding

For training, we define a reconstruction and a regularization losses.

For instance, the reconstruction loss may be the square error :

$$\mathcal{L}_{rec}(\mathbf{x}_n, \mathbf{v}_n) = ||\mathbf{x}_n - \sum_{k=1}^{K} a_{n,k} W_k||_2^2$$

And the regularization, the L-1 norm of the activation :

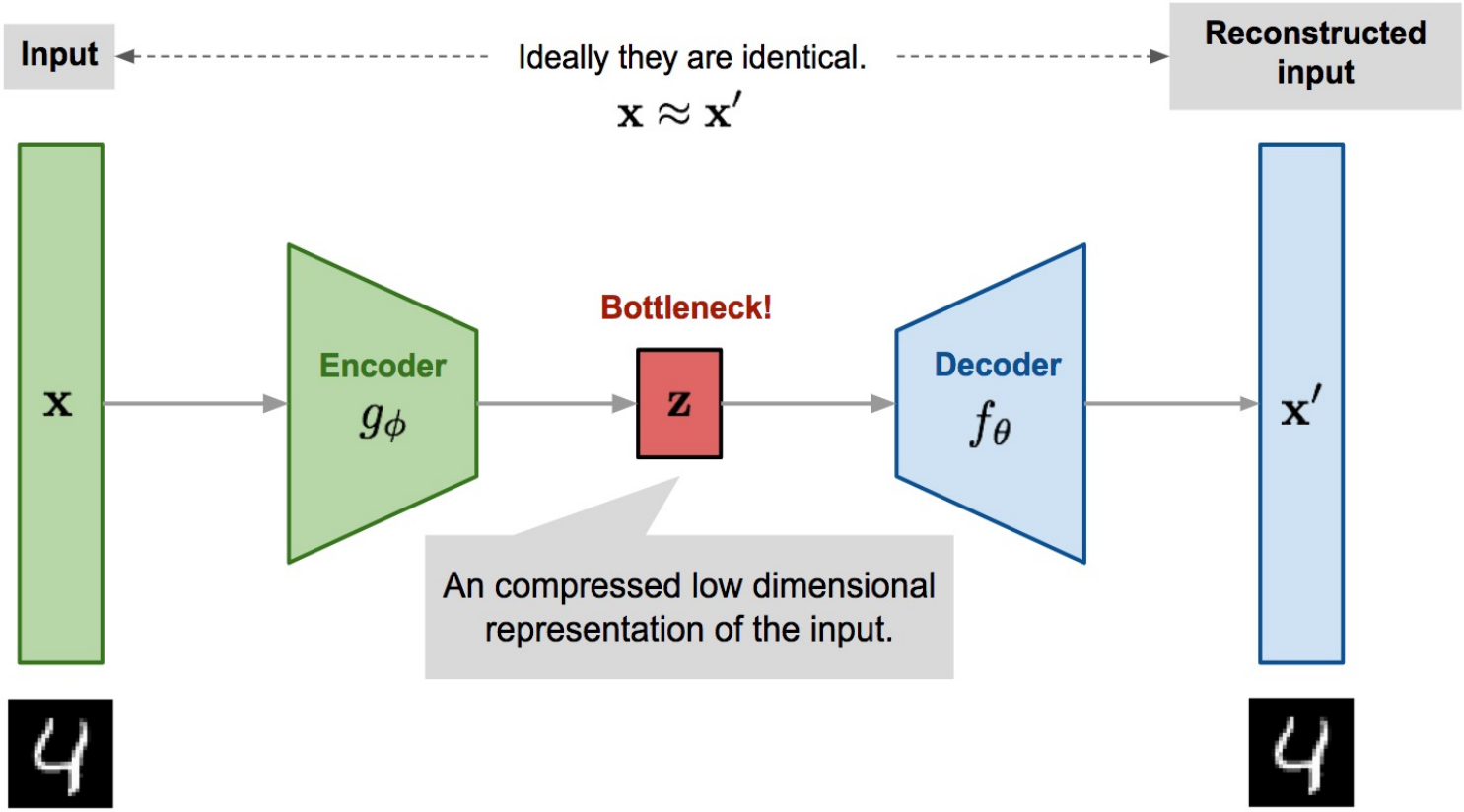
$$\mathcal{L}_{pen}(\mathbf{a}_n) = \sum_{k=1}^{K} |a_{n,k}|$$

So that the optimization is conducted to minimize the total loss :

$$\min_{\mathbf{a},\mathbf{W}} \sum_{n=1}^{N} \mathcal{L}_{rec}(\mathbf{x}_n, \mathbf{v}_n) + \mathcal{L}_{pen}(\mathbf{a}_n)$$

Applications

- Lossy compression & coding
- Dimensionality reduction, visualisation



Applications

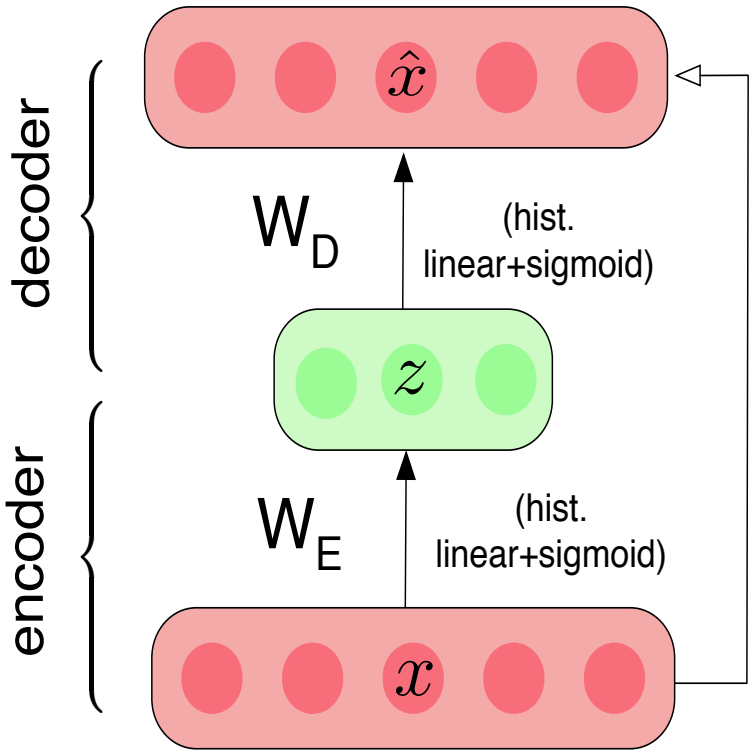
- Generating data

Deep fakes, etc..

SCIENCES SORBONNE UNIVERSITÉ

2. Auto-encoders (AE)

Definition



$\mathcal{L}_{AE}(x, \hat{x})$

(hist. MSE)

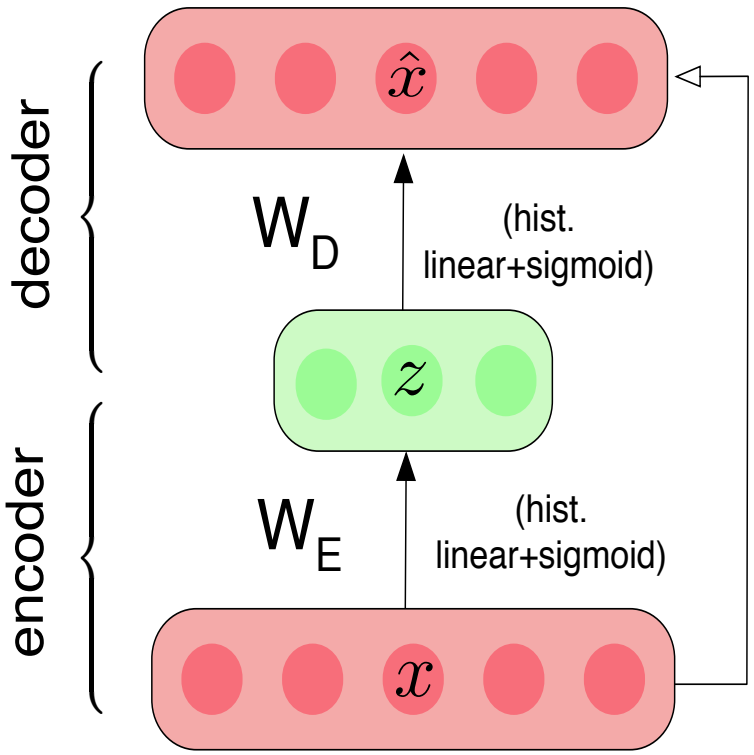
Definition

An AE is a symmetrical NN with one hidden layer

Principle

- One want to learn a latent code z
- generally of smaller dimension than the input dimension x
- From which we can best reconstruct the input

Definition



$\mathcal{L}_{AE}(x, \hat{x})$

(hist. MSE)

Architecture

An AE is composed of an encoder and a decoder modules

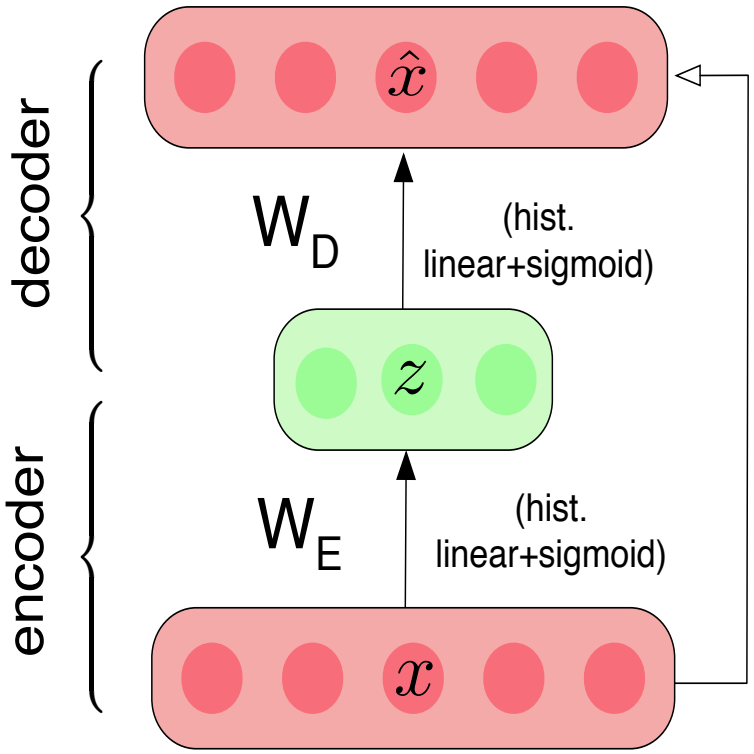
Encoder

The encoder compresses the input **x** into a latent code **z**

Decoder

The decoder reconstructs the input **x** from the latent code **z**

Learning



$\mathcal{L}_{AE}(x, \hat{x})$

(hist. MSE)

Objective

One wants to minimize the error between the original input and its reconstruction (typically, by using a mean square error)

Solution

Classic optimisation (SGD, ADAM, etc..)

Potential issue

How do we avoid learning the identity mapping?

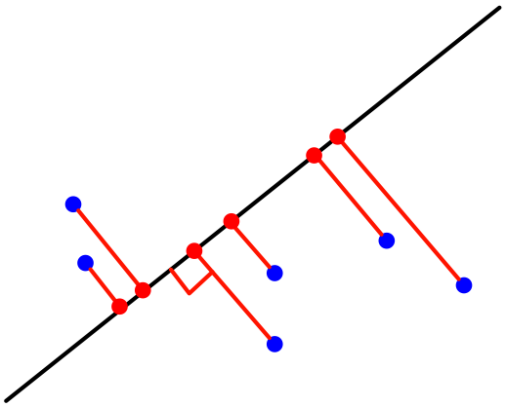
- The smaller dimension of z avoids this issue

Discussion

Linear auto-encoder

A linear AE linearly projects the input data x from a D-dimensional space to a K-dimensional latent subspace Z (generally with K<D)

By definition, the projection of **x** on Z is the point of Z minimizing the distance to **x**



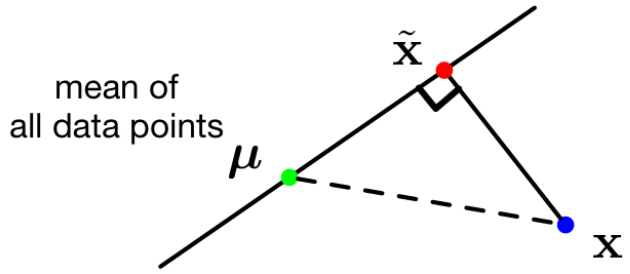
In the case of a linear AE, this projection is represented by the linear transforms U=Q et $V=Q^T$ where Q is an orthonormal basis of Z

Discussion

Linear auto-encoder

The linear AE must learn the sub-space which minimizes the quadratic distance between the data points **x** and their projection **x**

By application of the Pythagorean theorem, one can show that this solution is equivalent to maximizing variance of the projections



There is a well-known algorithm for this linear problem: principal component analysis (PCA)

➢ No need to train a NN !!!

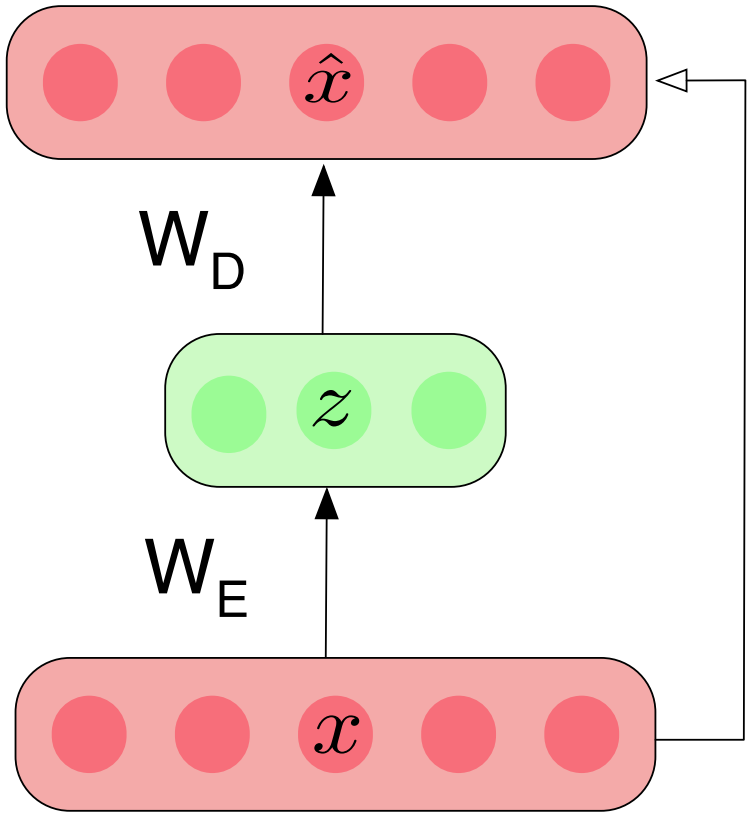
AE and semi-supervised learning

- The AE learns a compressed representation z of the data

- Completely unsupervised!

- Potentially from a large amount of **unlabelled data** !

➢ Representation learning

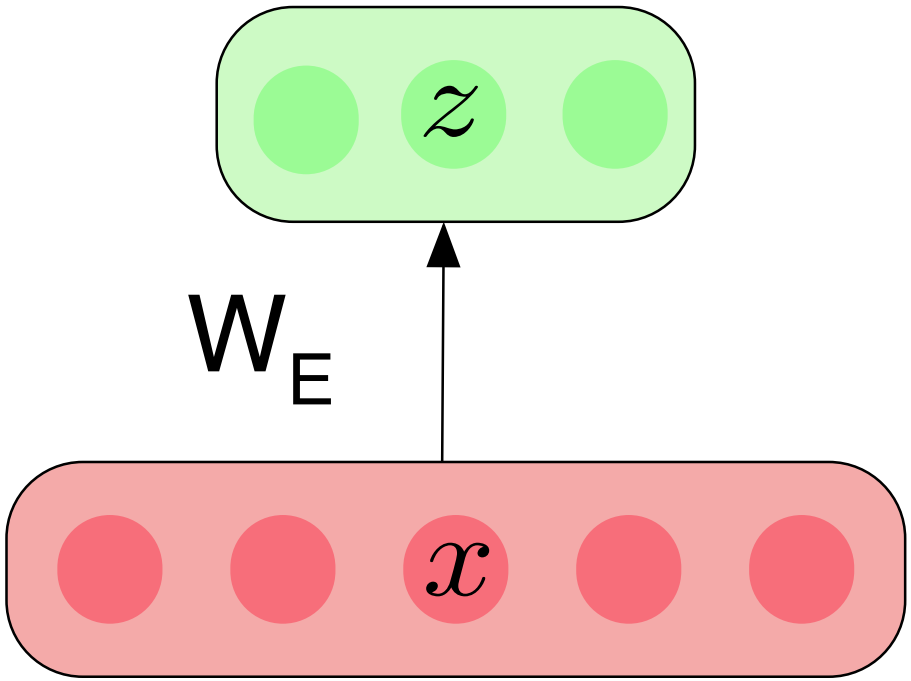


AE and semi-supervised learning

- Can this representation helps for supervised tasks?

- Once AE training is finished

➢ On can keep its encoder part (and the corresponding weights W_E)



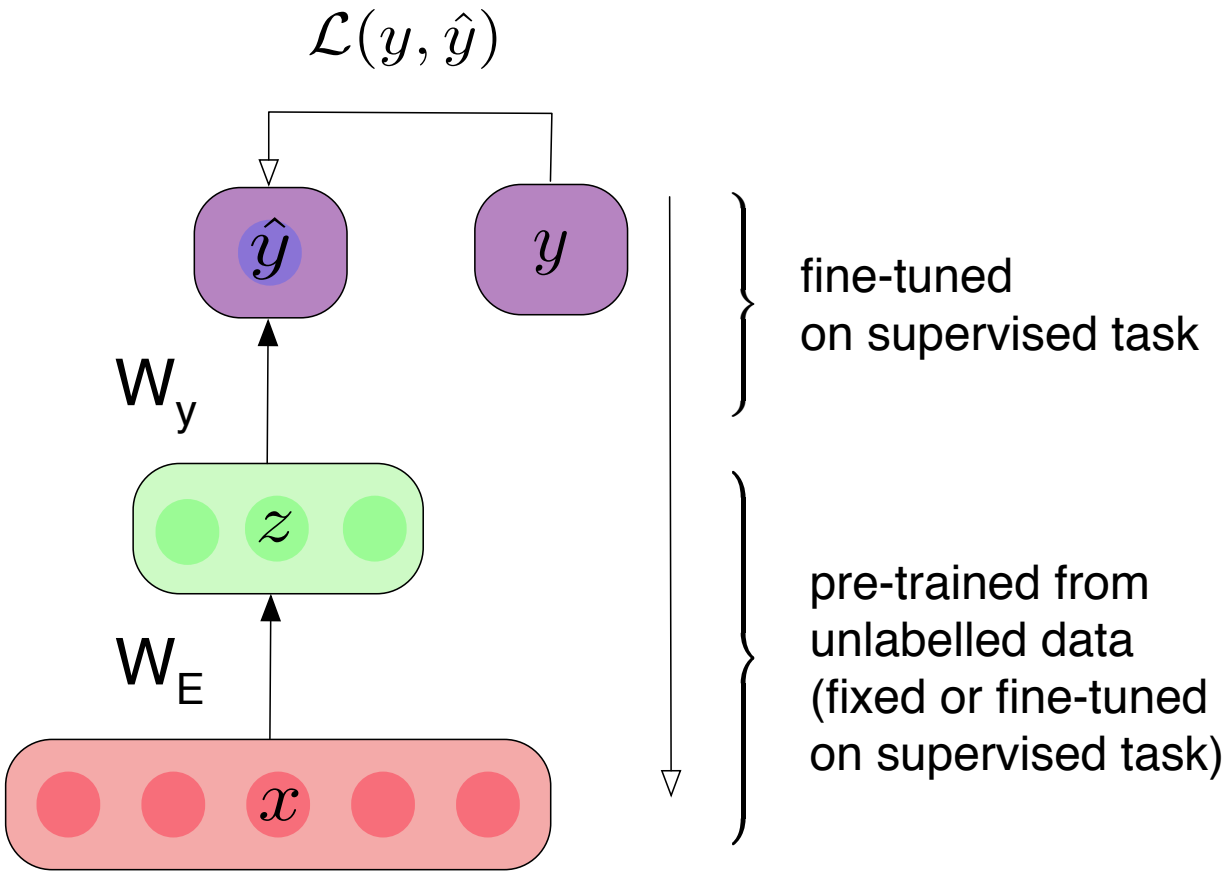
AE and semi-supervised learning

- Once AE training is finished (aka **pre-training**)

➢ On can keep its encoder part (and the corresponding weights W_E)

➢ To initiate a supervised learning phase

This phase is aka: **fine-tuning**



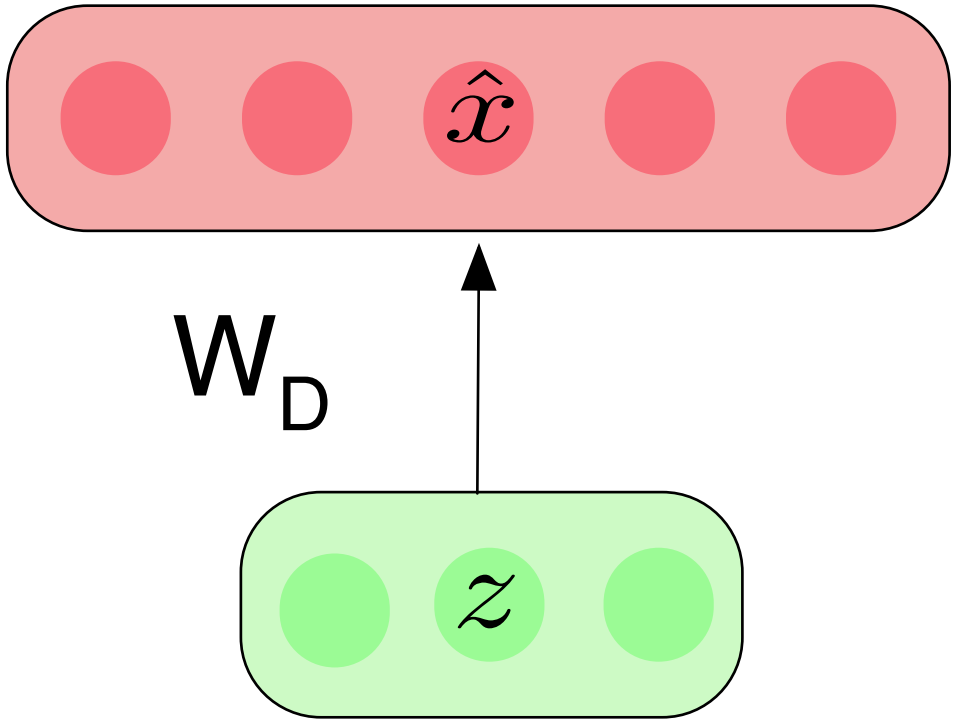
AE as a generative model

- Conversely, one can keep its **decoder** part

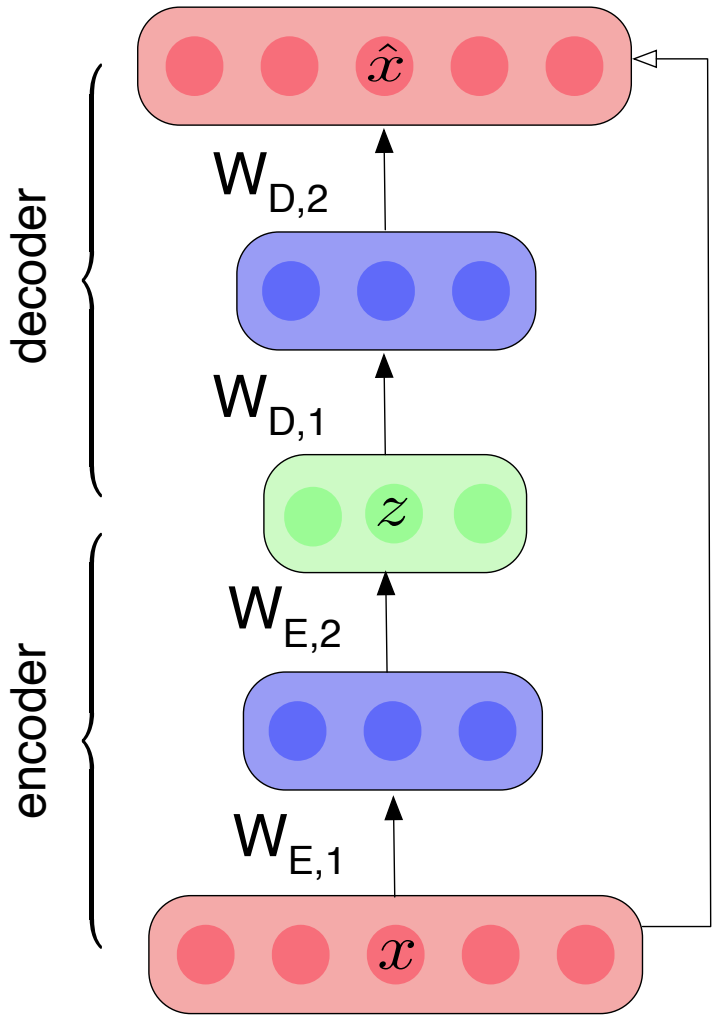
- Generative model

- How to chose **z** so as to generate realistic data

- + with desired specifications?



From shallow to deep AE



$\mathcal{L}_{AE}(x, \hat{x})$

(hist. MSE)

Definition

- Same principle as AE
- But with more hidden layers (and non-linearities)

Architecture

- Generally symmetrical
- But not necessarily
- ➢ Conditional AE: encoding and decoding not always with same complexity

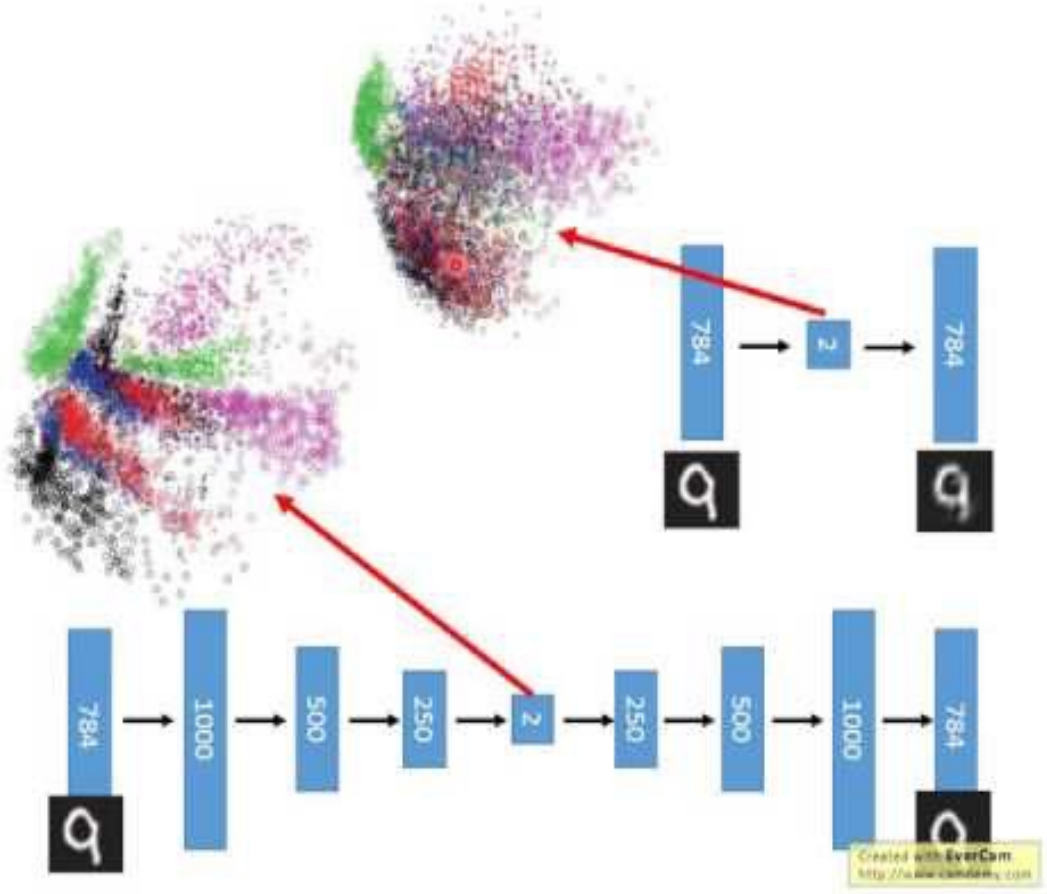
Short history

Origine

[Hinton and Salakhutdinov, 2006]

Deeper is better

Better digits separation with deeper NN



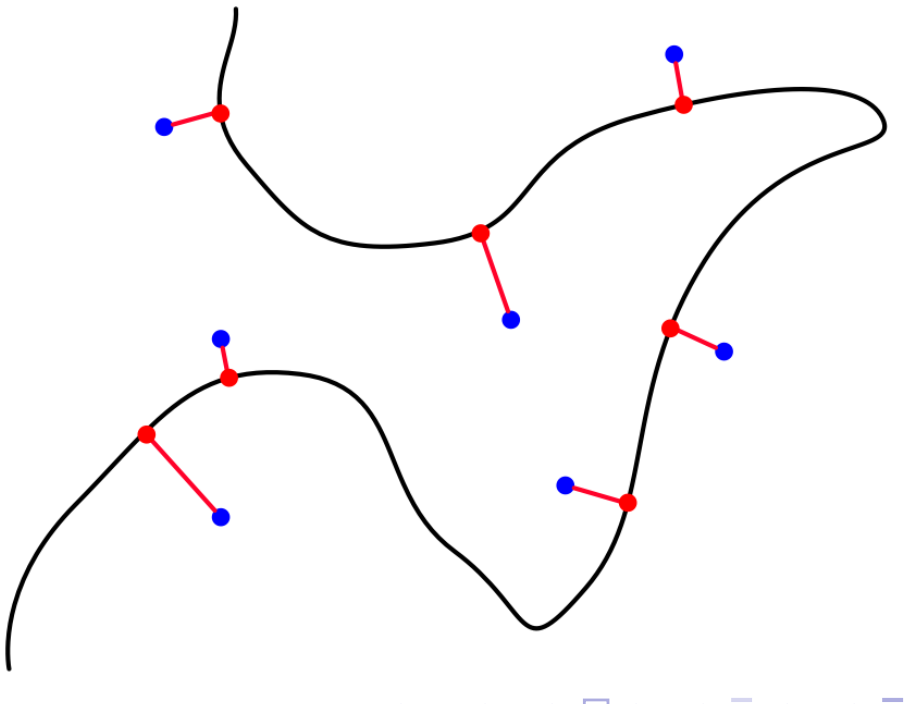
Discussion

Deep (and non-linear) AE

In the case of a multi-layer AE (with non-linearities), the projection is not on a sub-space but on a **manifold**.

This manifold is the **image of the encoder**

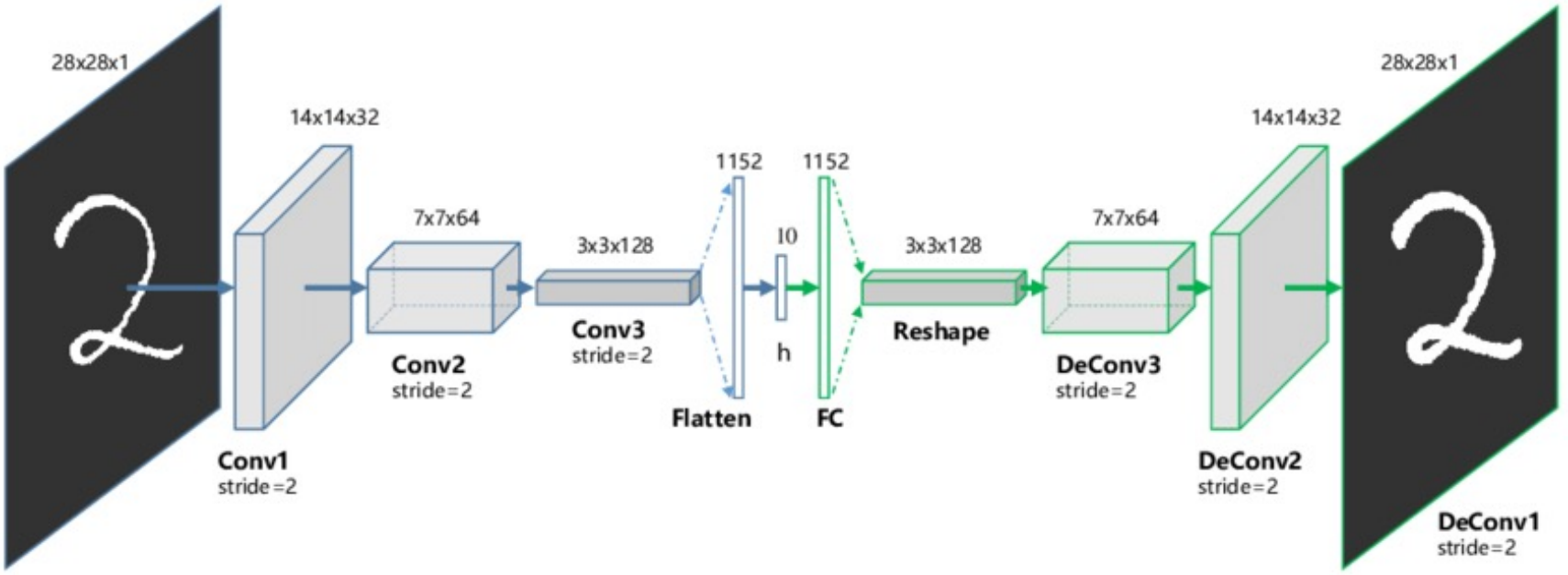
- Non-linear dimensionality reduction



Architectures

AE can be implemented with all existing NN:
FC, CNN, RNN, etc...

Families of AE :
- Sparse coding
- Denoising AE
- Variational AE
- etc...



Regularization

Learning invariances

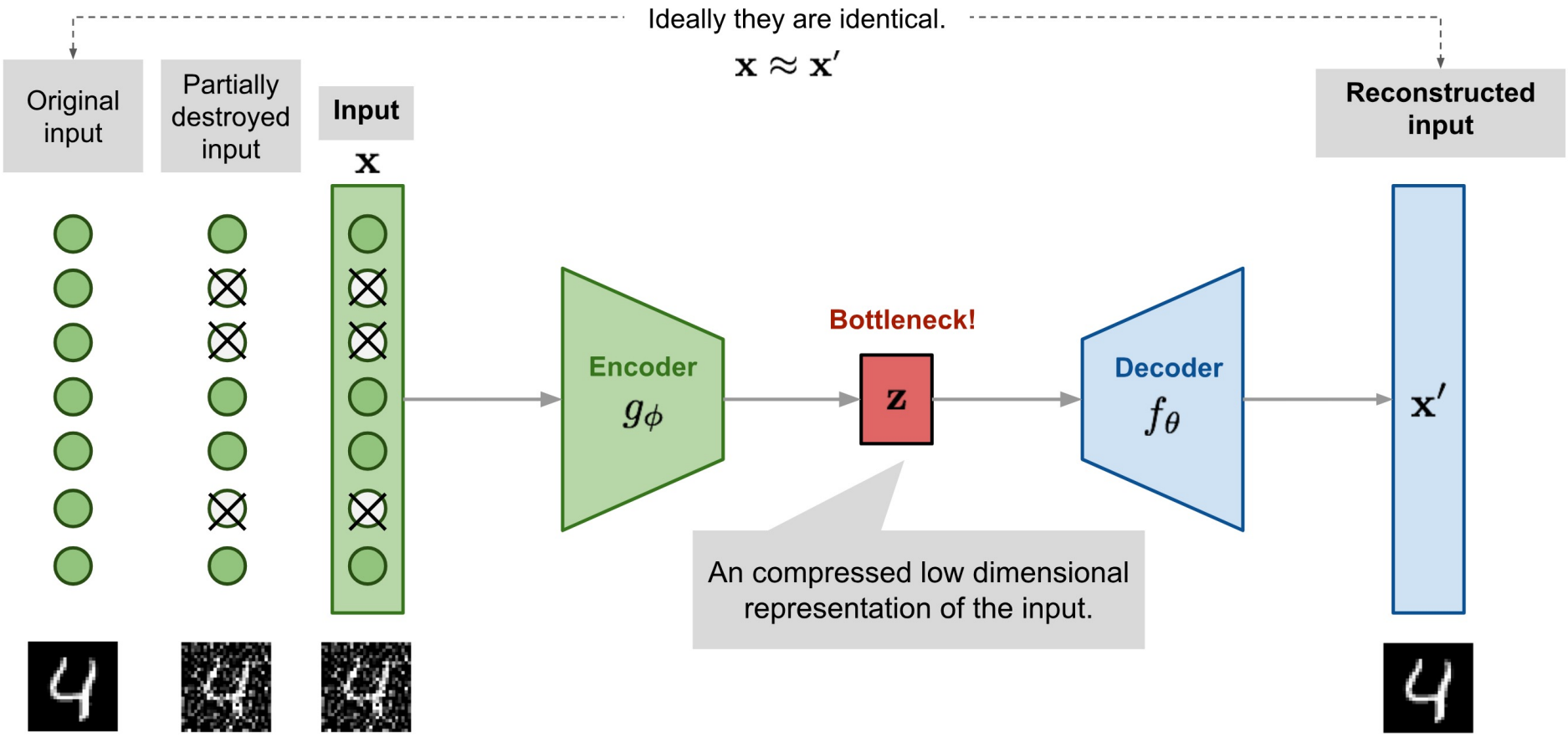
The latent code **z** can be subject to some invariance properties

For instance, invariance to noise

Denoising AE

[Vincent, Larochelle, Bengio, Manzagol, 2008]

Regularization



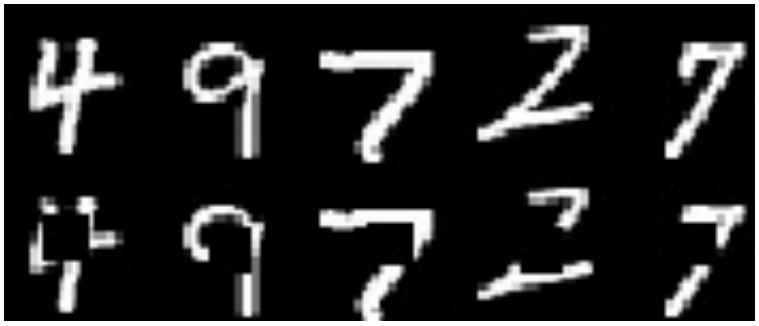
Regularization

Principle

The network tries to reconstruct a clean data from a corrupted version

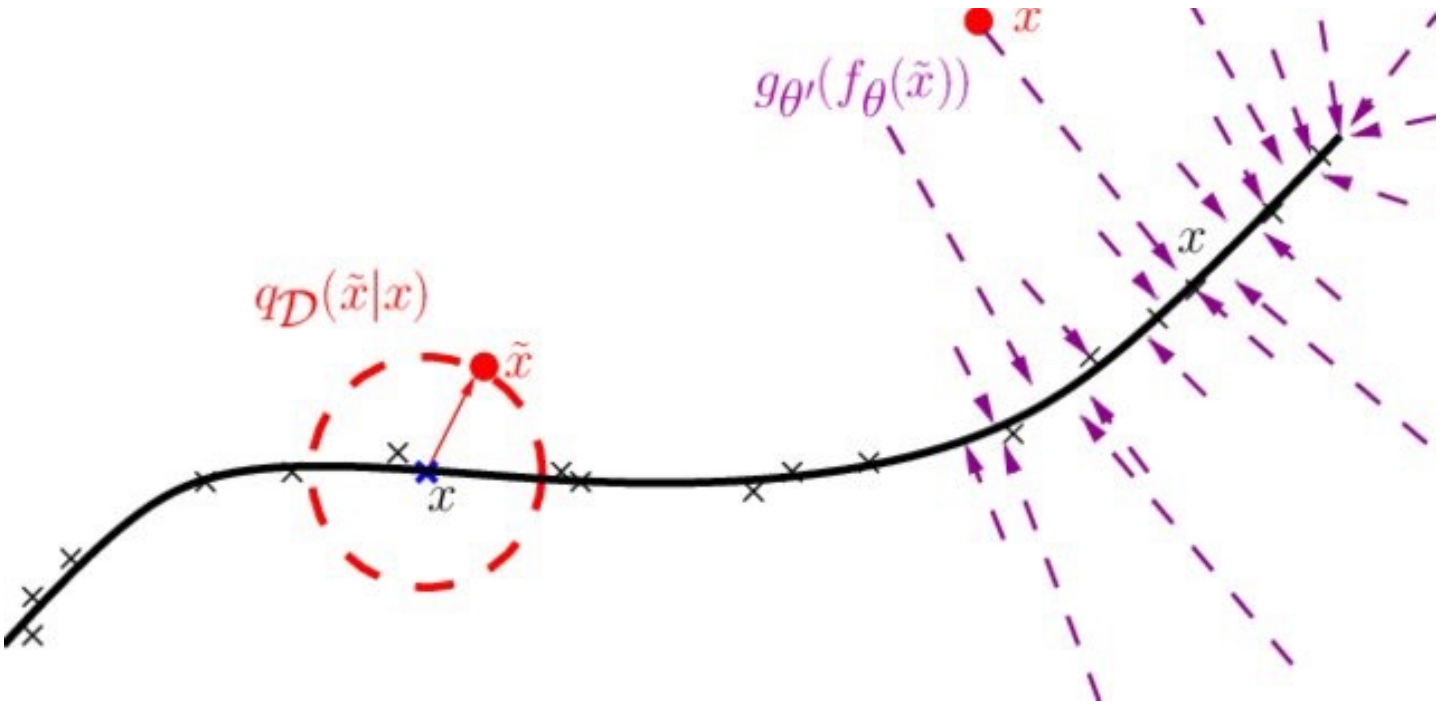
Some examples

- Additive / multiplicative noise
- Binary masks (proportion of pixels set to 0, masked region)
- Among others ....



Regularization

- Function q_D is a corrupting process displacing original data **x** from the manifold as x
- Encoder f and decoder g learn to project x back onto the manifold



Source : Goodfellow, I., Bengio, Y. & Courville, A. Deep Learning. (MIT Press, 2016).

Regularization

Learning invariances

- Other manipulation can be applied on the original input data (e.g., rotation, translation, scaling, etc…) depending on the desired **invariances properties**

Interpretation

- This encourages the latent code z to contain only information about the digit
- But not its actual position in space, etc…

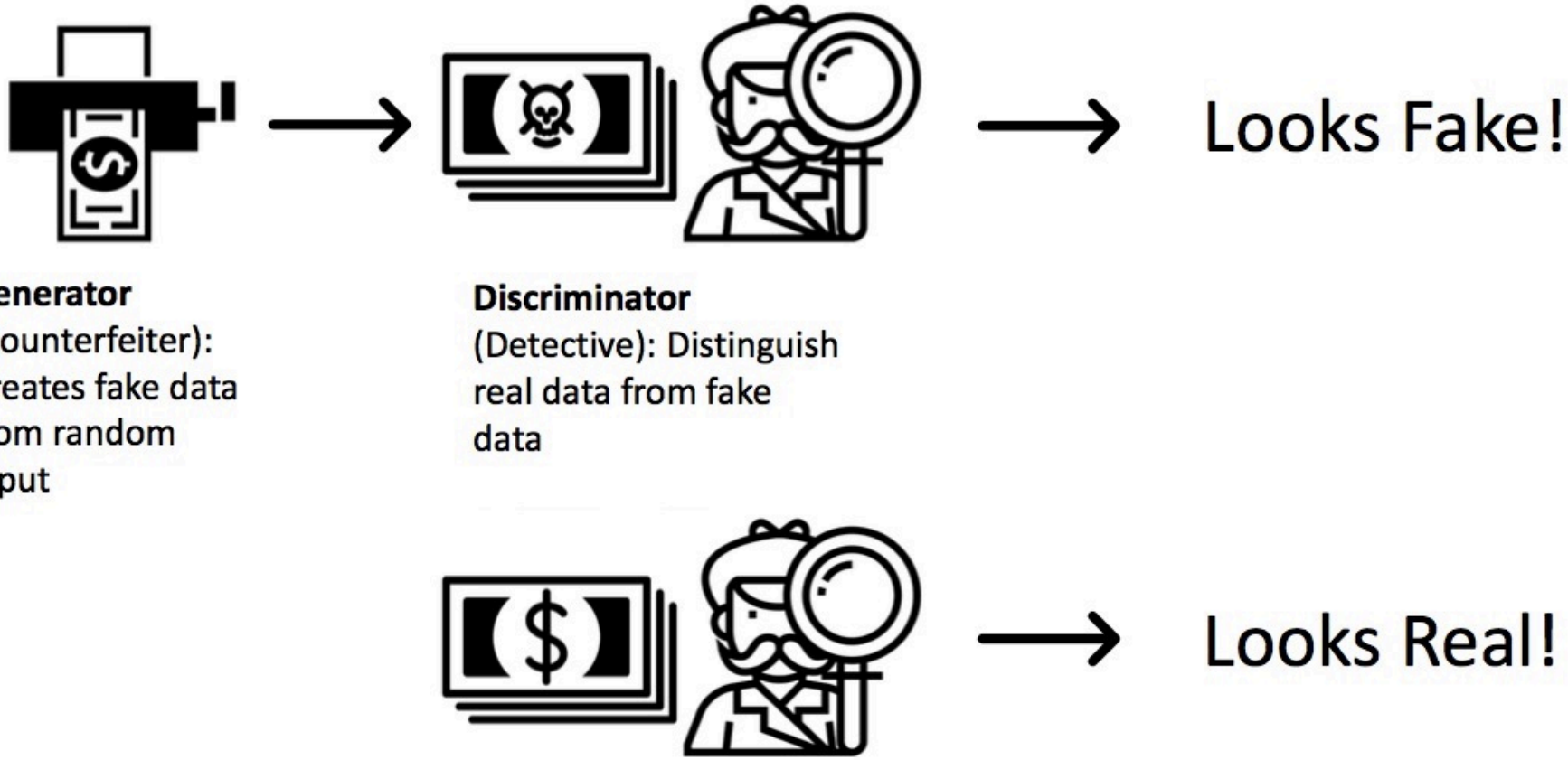
➢ Bottleneck effect

SCIENCES SORBONNE UNIVERSITÉ

4. Generative adversarial networks (GAN)

Idea

(Goodfellow et al., 2014) as inspired by game theory!!!



Generator (Counterfeiter): Creates fake data from random input

Discriminator (Detective): Distinguish real data from fake data

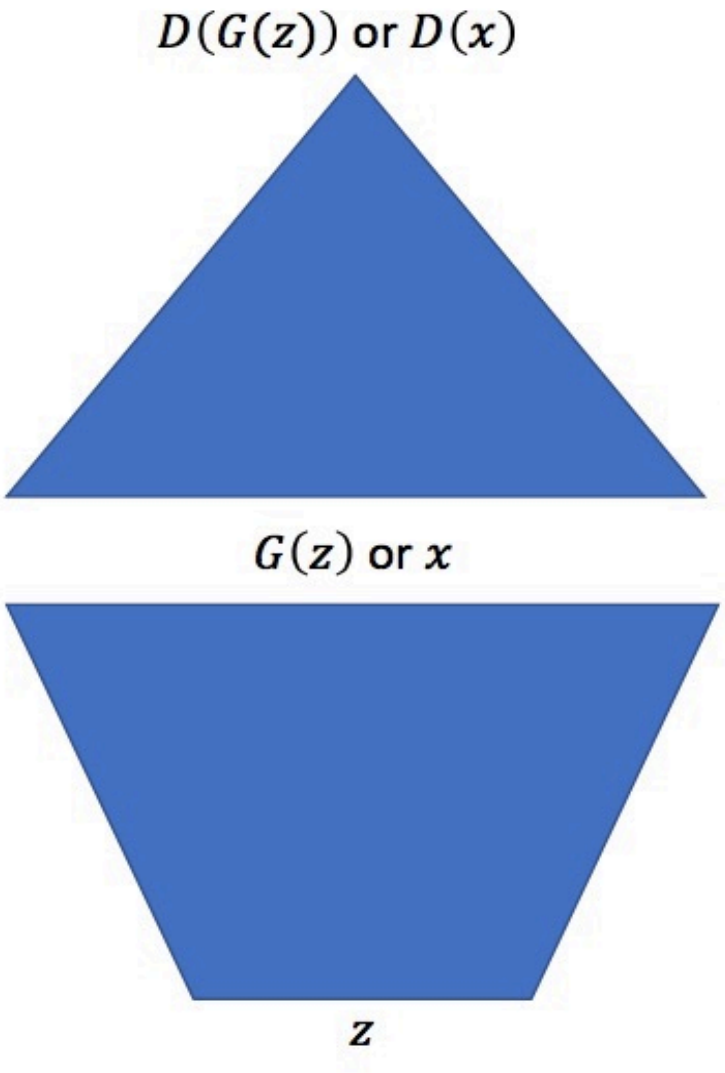
Idea

Two networks with **adversarial objectives** are competing with each other in a game

- Generator: tries to produce fake data that look real
- Discriminator: tries to distinguish fake data from real data

Together, they helps learning the unknown distribution of data

Formalization



Generator input

z: random latent code (drawn from random or uniform distribution)

Generator

From z generate fake data G(z)

Discriminator

From real data x or fake data G(z), apply decision operator D (is real or is fake)

Formalization

Generator: $G\left(z, \theta^{(G)}\right)$
A differentiable function, G (here having parameters $\theta^{(G)}$), mapping from the latent space, $\mathbb{R}^L$, to the data space, $\mathbb{R}^M$

Discriminator: $D\left(x, \theta^{(D)}\right)$
A differentiable function, D (here having parameters $\theta^{(D)}$), mapping from the data space, $\mathbb{R}^M$, to a scalar between 0 and 1 representing the probability that the data is real

GAN & game theory

G and D are competing in a game with 2 players

G and D only control their own parameters (otherwise this would be cheating!!!)

Each player tries to maximise its chance of success and to minimize the chance of success of its adversary

- Minmax optimization

Loss functions

The loss function for the discriminator D is defined as:

$$L^{(D)}(\theta_D, \theta_G) = -\frac{1}{2}E(\log(D(x))) - -\frac{1}{2}E(\log(1 - D(G(z))))$$

The decision function D(x) is defined as follows:

- If x is real, D(x)=1
- Otherwise (x is fake), D(x)=0

First term: ability to classify positive examples (real data)
If x is real data, then D(x)=1, and log(D(x)) = 0

Second term: ability to classify negative examples
If data is fake data G(z), then D(G(z))=0, and log(1-D(G(z)))=0

Loss functions

The loss function for the generator G is defined adversariarily to the one of the discriminator D, as:

$$L^{(G)}(\theta_D, \theta_G) = -L^{(D)}(\theta_D, \theta_G)$$

Only need to optimize the second term in the equation from the previous slide (the one involving G)

Min-max optimization

$$\min_G \max_D \left(-L^{(D)}(\theta_D, \theta_G)\right)$$

- The discriminator D tries to minimize $L^{(D)}$ (alt. maximize $-L^{(D)}$), the classification loss between real and fake data
- In turn, the generator G tries to minimize this same loss $-L^{(D)}$

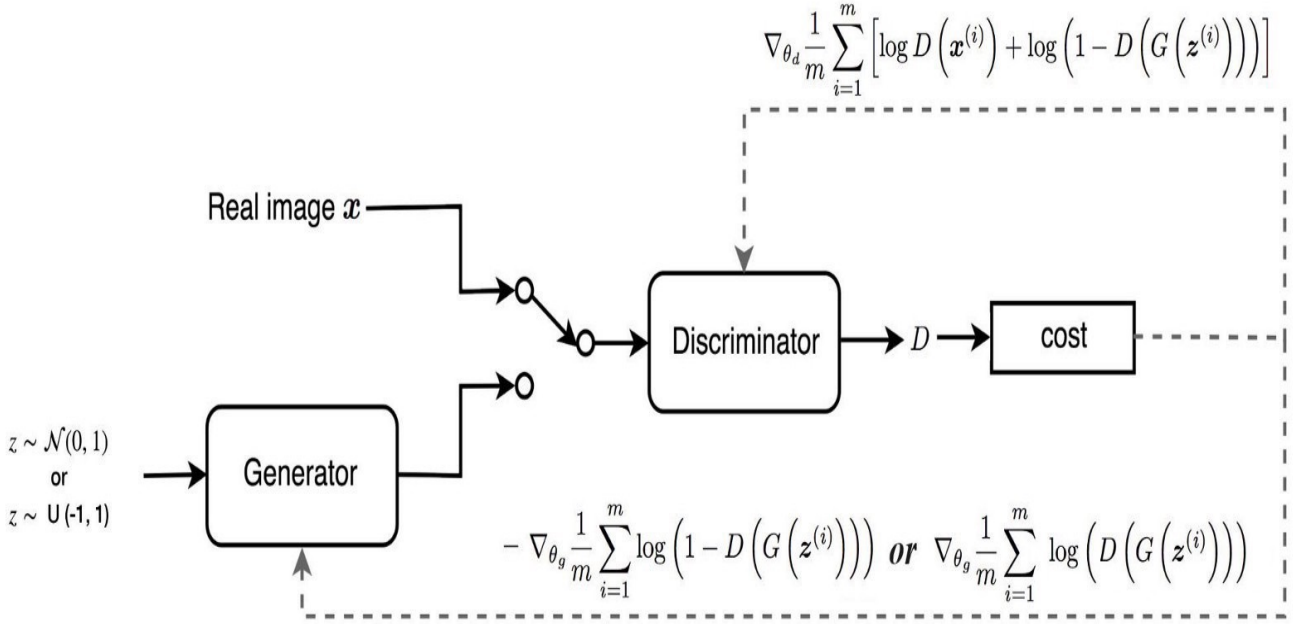
➢ This is achieved through alternated optimization of G and D

Alternated optimization

A GAN is a good example of a non-trivial neural network:

- Multiple losses are defined and used
- **Each loss affects only a subset of the weights**

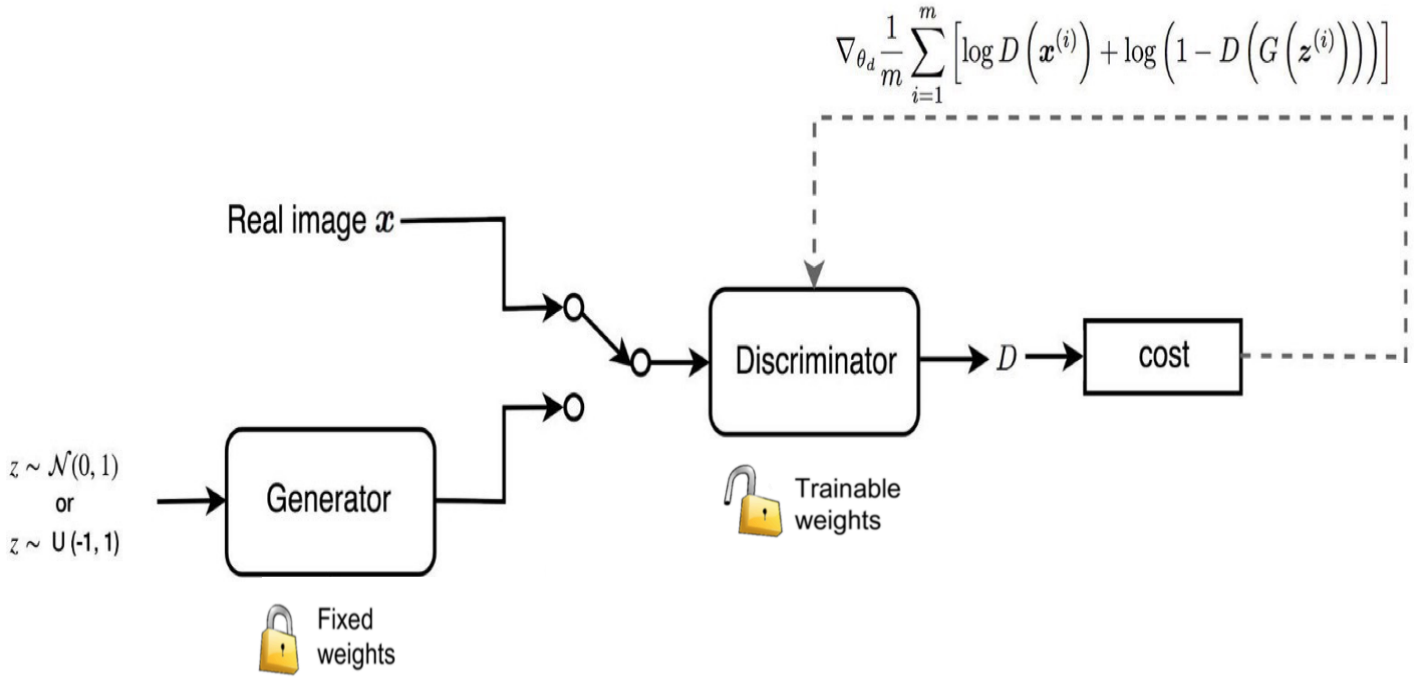
- G and D are adversarial: G is optimized w/o the capacity to modify the weights of D (and, reciprocally)
- G et D are optimized **alternatively**



Alternated optimization

Optimization for discriminator D

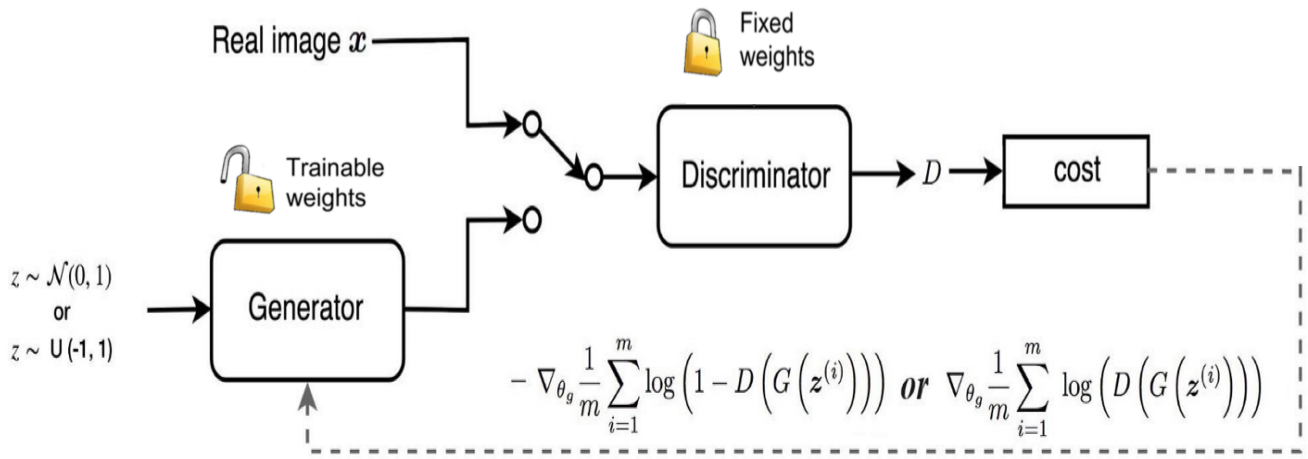
- The weights of the generator G are fixed (i.e., not accessible for modification)
- Generate data with G by drawing from the z distribution
- Calculate loss and gradient of the discriminator D (with real and fake data)
- Update the weights of the discriminator D



Alternated optimization

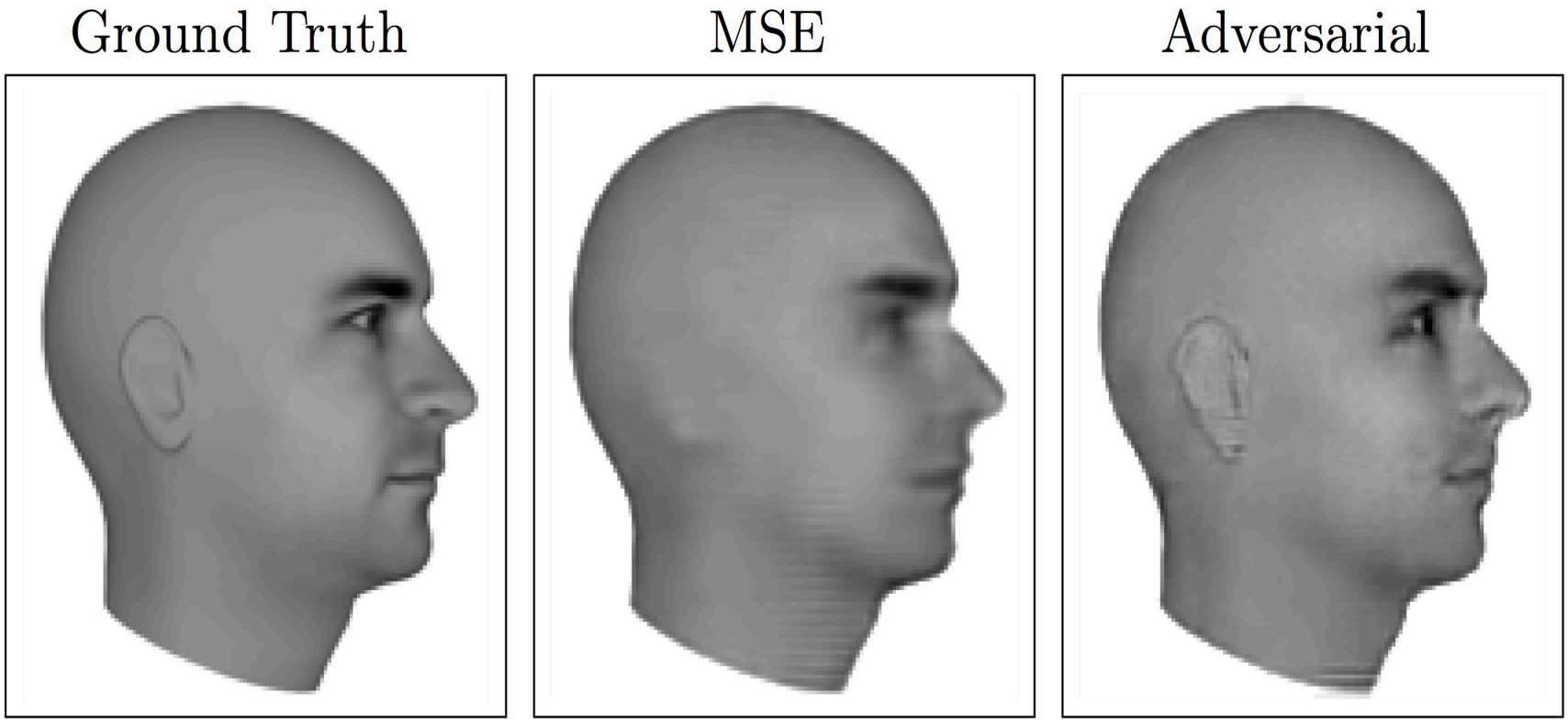
Optimization for generator G

- The weights of the discriminator D are fixed (i.e., not accessible for modification)
- Generate data with G by drawing from the z distribution
- Calculate loss and gradient of the generator G (only with real data)
- Update the weights of the generator G



Architectures: Vanilla GAN

MSE AE vs. GAN



Architectures: Vanilla GAN

Photo-realistic face generation by NVIDIA

GAN and beyond

Generative adversarial networks are now well-established

Many variants and advances:

- Loss :
  - LS-GAN,
  - Wasserstein GAN,
  - etc...
- Architectures :
  - GAN with **cycle-consistency** (see after),
  - VE-GAN using **Variational Auto-Encoder**,
  - Fader networks using adversarial learning (see after),
  - etc...

Known issues with GAN

In practice, GAN often can suffer from **stability issues** during learning

The main reason is that it is hard to regulate the learning of the two networks:

- One network may learn faster than the other,
- Thus, making impossible to the other to keep on improving

Known consequences are (but not limited to):

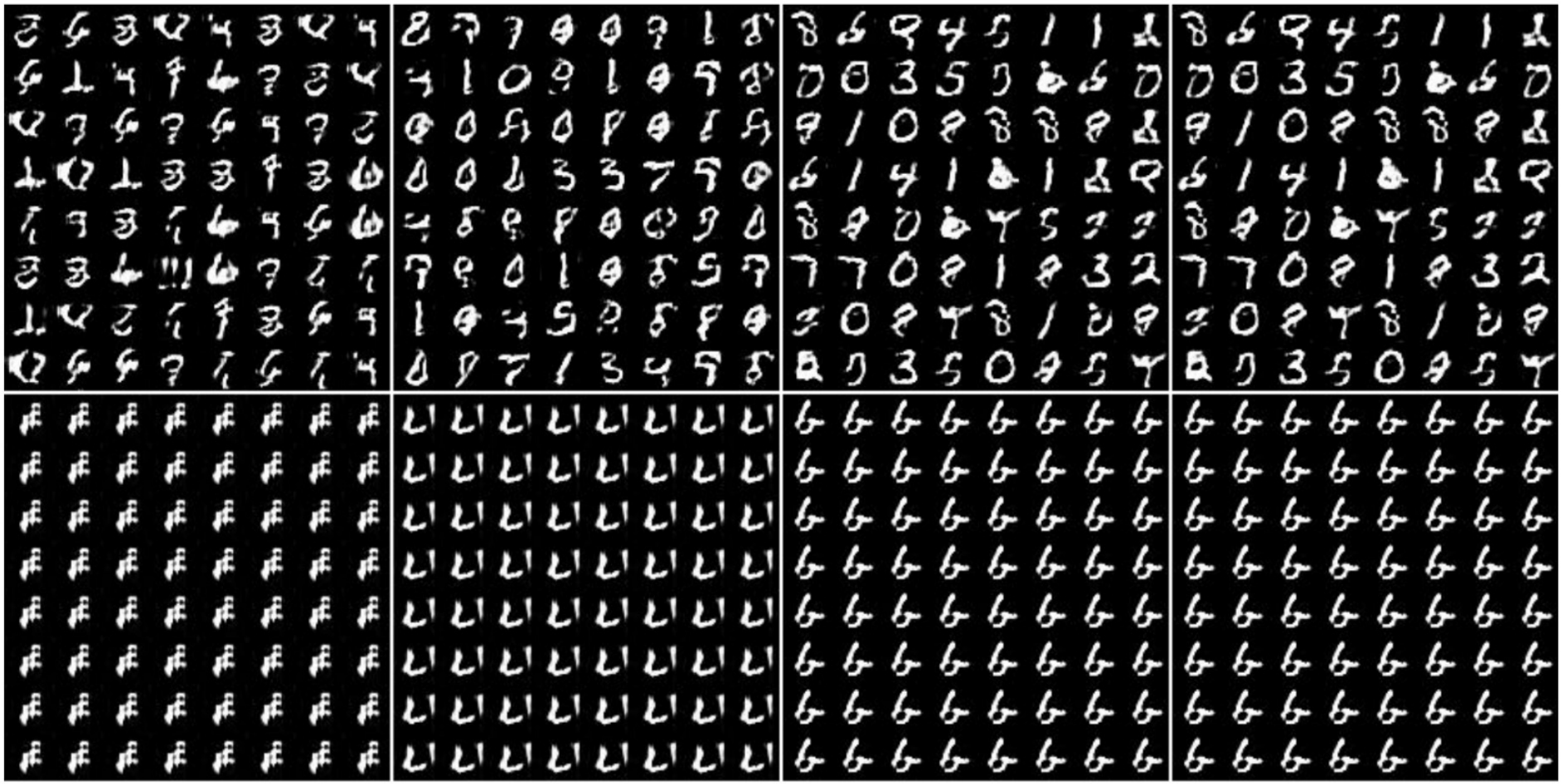
- **Oscillation** A GAN oscillates and fails to converge: each network take the advantage alternatively in an endless loop

- **Vanishing gradient** The discriminator takes the advantage over the generator, so that the generator does have the ability to fake data

- **Mode collapse** The generator converges to a – very – limited number of modes

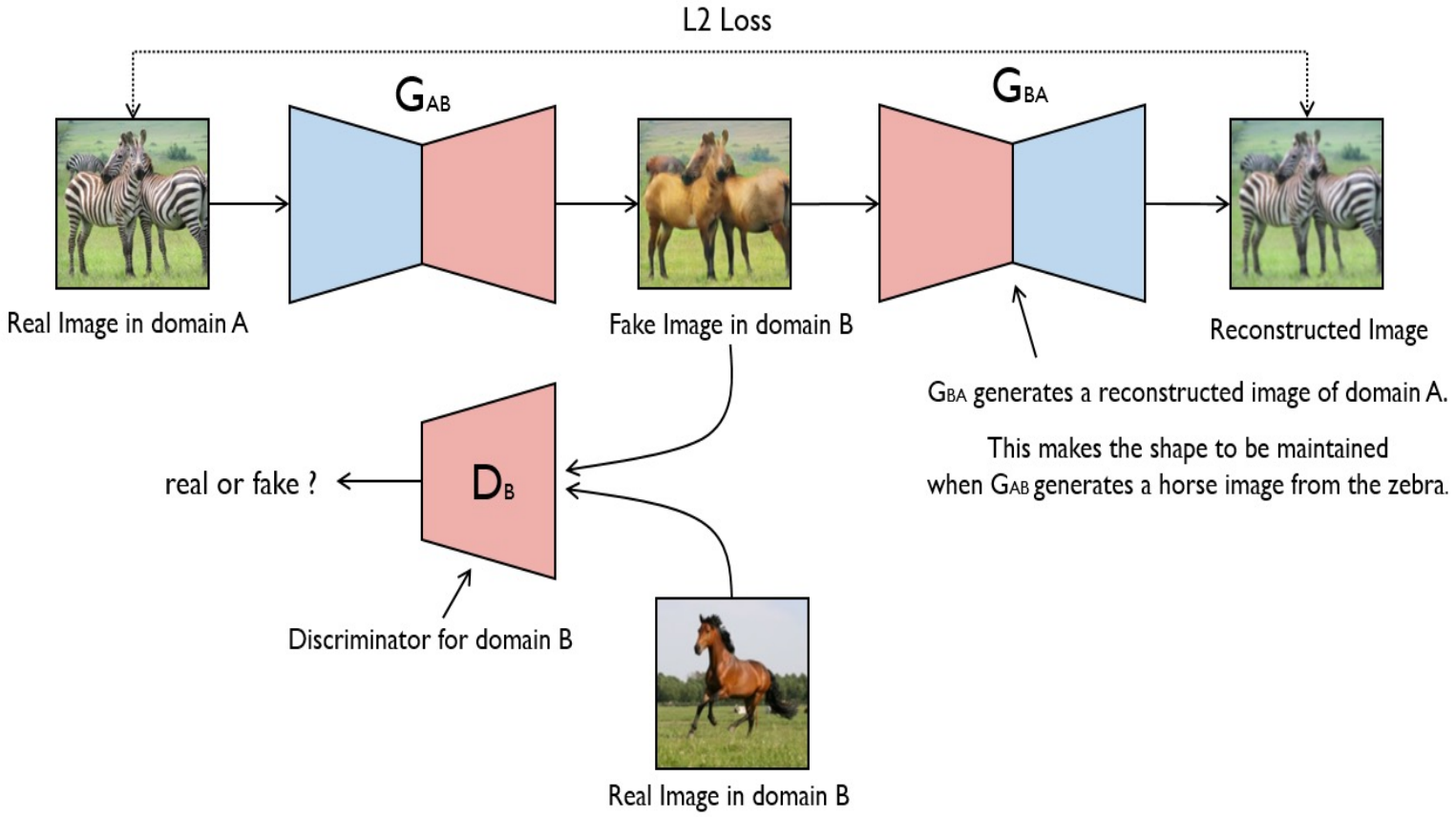
Known issues with GAN

Illustration of the « collapse mode »



Architectures Cycle-GAN

Style transfer and cross-domain translation from unpaired data



Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros (2017)
Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,
in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2242–2251.

Architectures Cycle-GAN

Adversarial loss between domains X and Y

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

G (or $G_{X->Y}$): **generator** which maps data sampled from X to data sampled from Y
D_Y: **discriminator** which distinguishes between real data sampled from Y and fake data sampled from $G_{X->Y}$

Conversely, an adversarial loss is defined the other way around between Y and X, with generator F (or $G_{Y->X}$) and discriminator D_Y

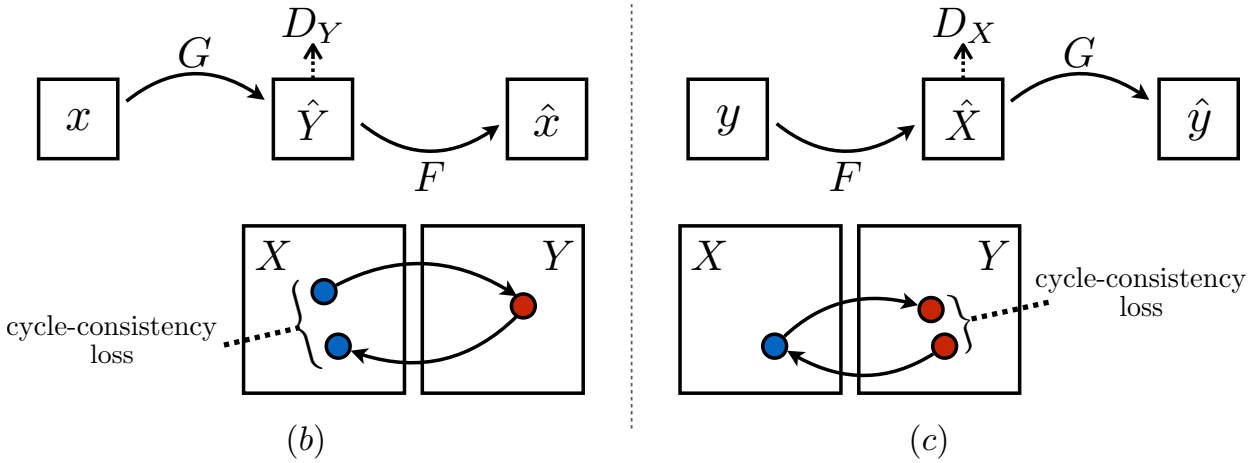
Architectures Cycle-GAN

Cycle-consistency loss

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) =& \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

The cycle-consistency encourages that:

- Forward consistency: the composition of F and G reconstructs the original input data x from X
- Backward consistency: the composition of G and F reconstructs the original input data y from Y



Architectures
Cycle-GAN

Total loss

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

Optimization

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

Two adversarial auto-encoders of domain X and Y, with the specific constraint that the latent code of each auto-encoder is the translation of the original domain to the target domain

In image-to-image translation, the cycle-consistency presents the empirical property:

- to **preserve the structure** of the original image
- while **translating the details** of the target image

SCIENCES SORBONNE UNIVERSITÉ

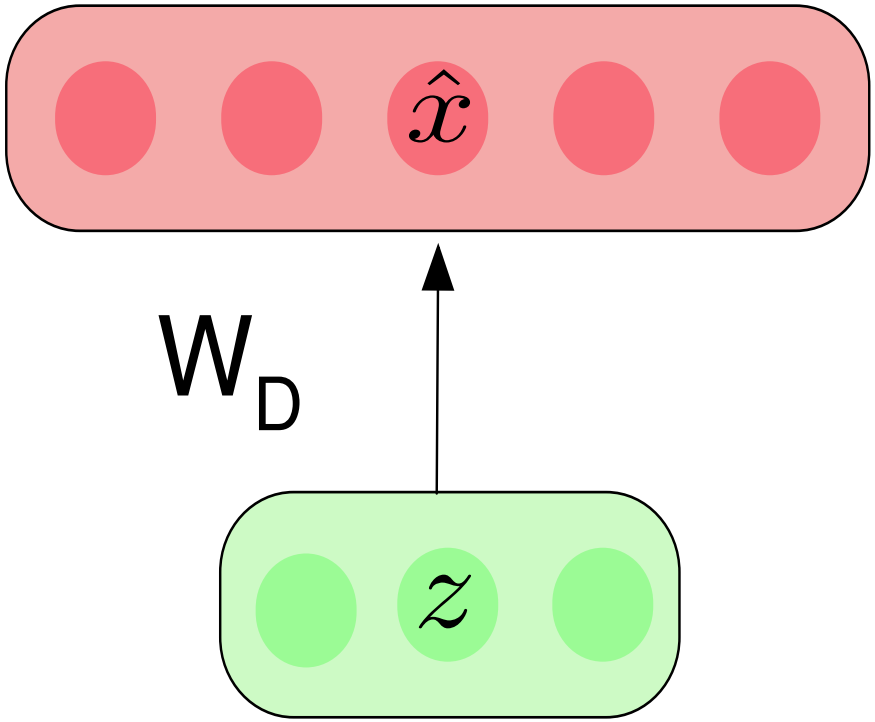
5. Conditional Auto-Encoders

AE as a generative model

- Conversely, one can keep its **decoder** part

- Generative model

- How to chose **z** so as to generate realistic data



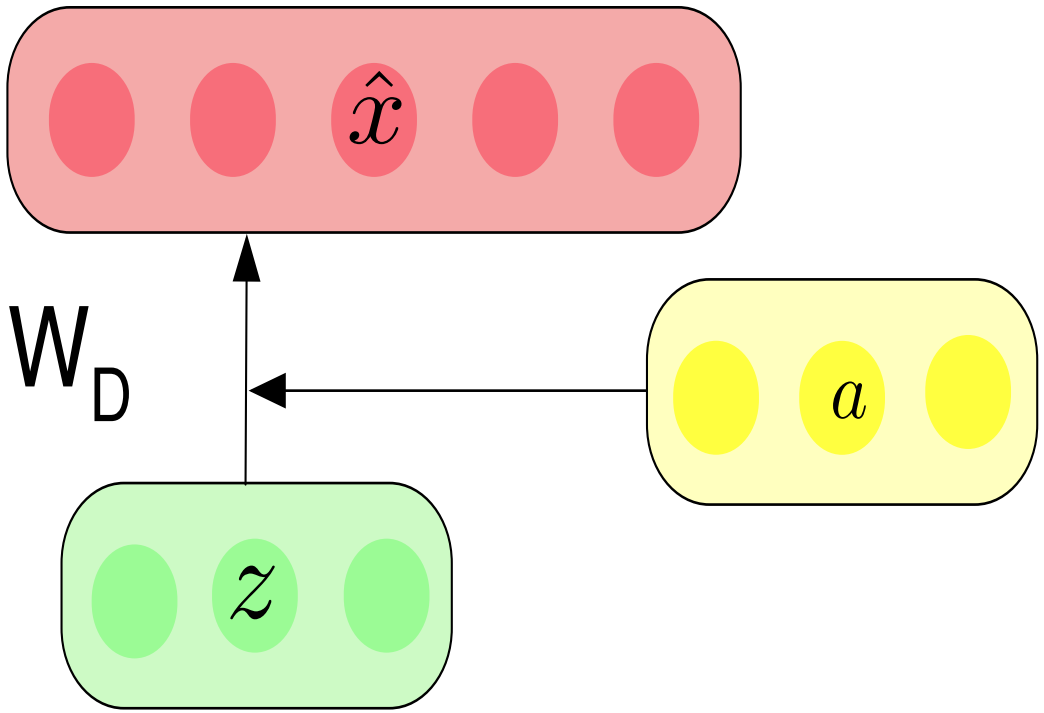
AE as a generative model

- Conversely, one can keep its **decoder** part

- Generative model

- How to chose **z** so as to generate realistic data

- + with desired specifications from code a?



Discussion

Conditional AE belongs to the family of **generative models**

- Decoder **D** generates data conditionally to latent code **z**

GAN

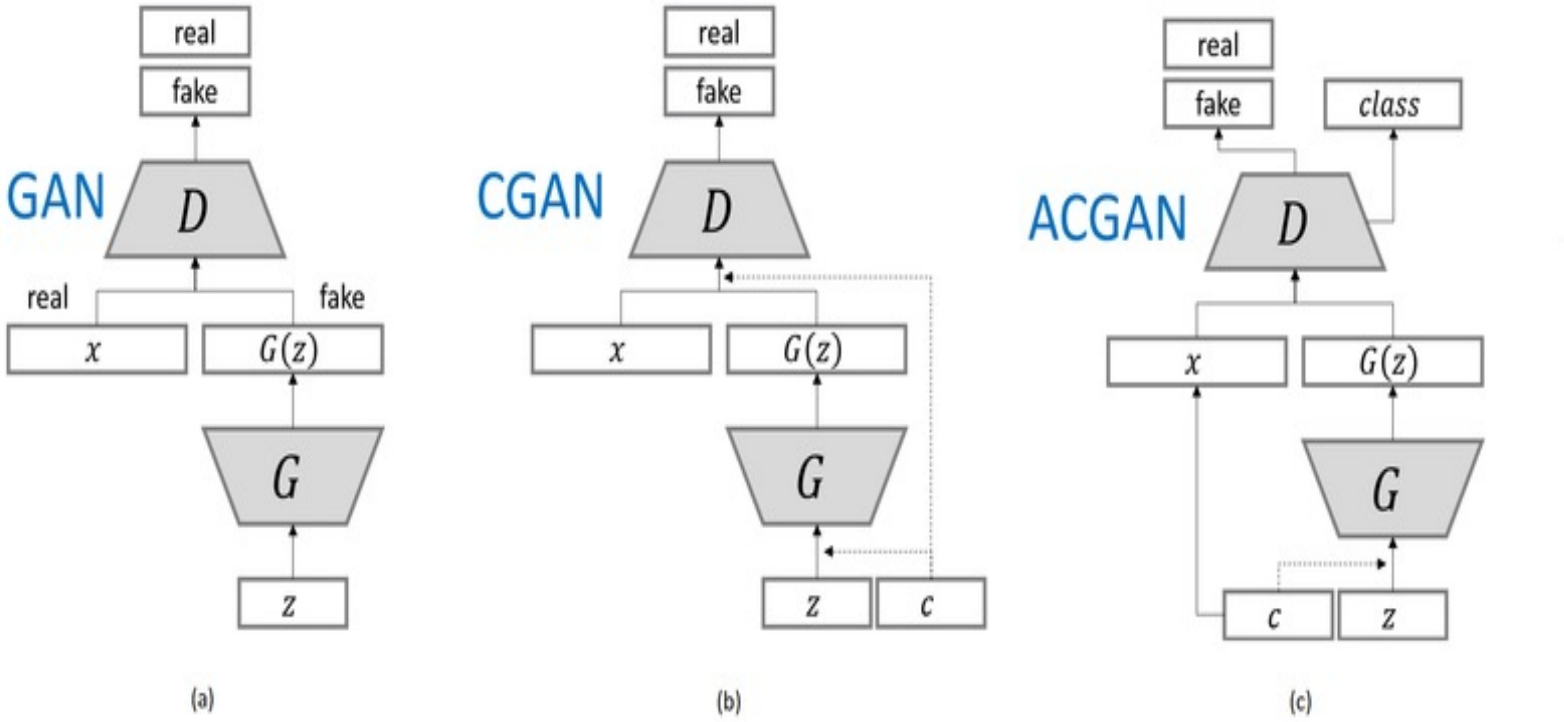
- In the case of a GAN, latent code **z** is drawn from a random distribution
- One does not control the generation rendering (except that it looks realistic and is derived from the distribution of original data)

AE

- In the case of an AE, latent code **z** results from the encoding by E of an input data **x**, such as **z** = E(**x**)
- The generation is limited to reconstruction, and the latent code z is not easily controllable

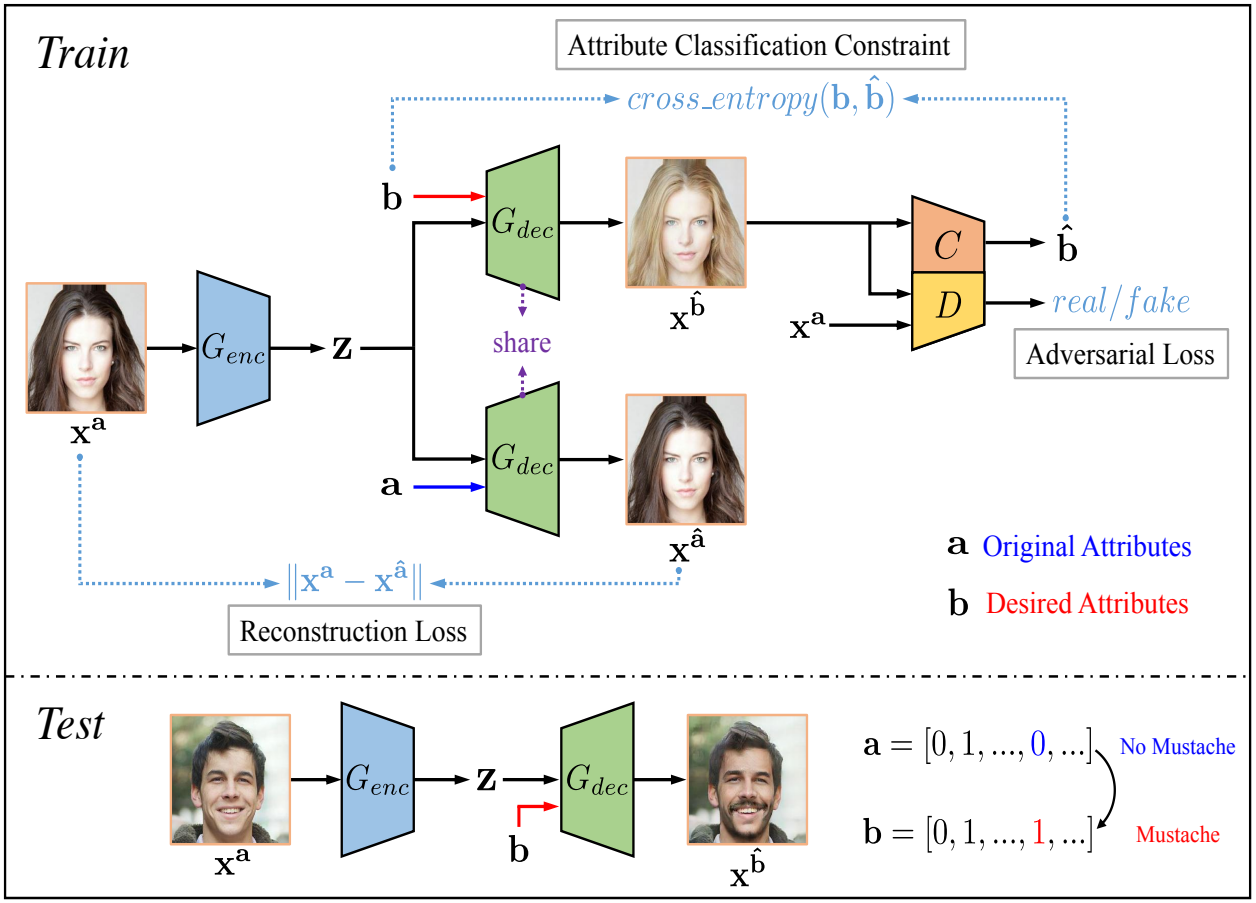
➢ How to add an extra conditioning variable and to make this conditioning effective?

GAN, Conditional-GAN, and Auxiliary-classifier GAN



(A. Odena, C. Olah, and B. Shlens, 2019)
Conditional image synthesis with auxiliary classifier GANs,
in International Conference on Machine Learning, 70, 2017.

Att-GAN



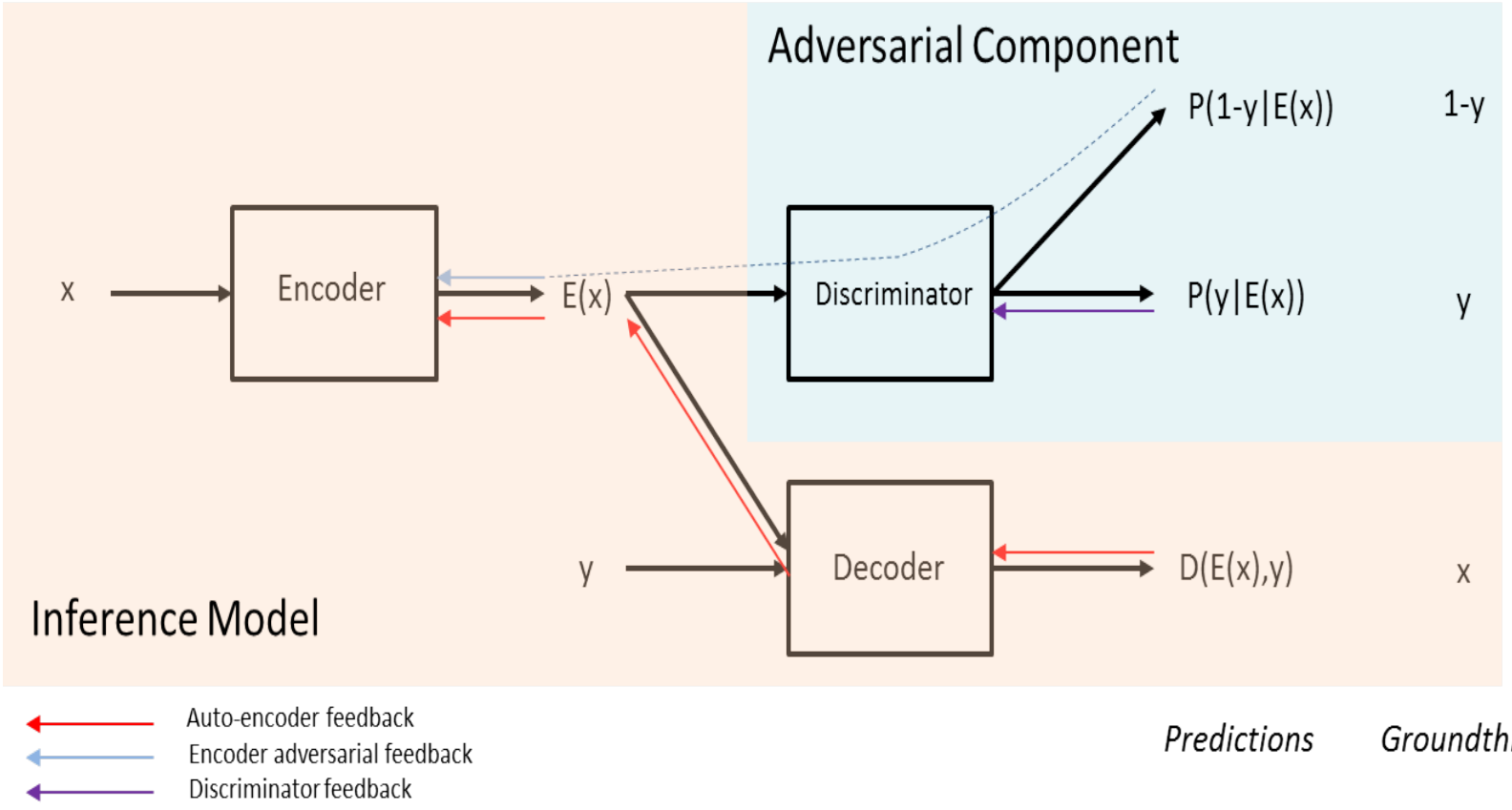
(He et al., 2019)
AttGAN: Facial Attribute Editing by Only Changing What You Want,
in IEEE Transactions on Image Processing, 28(11), 2019.

Fader network

Examples of manipulations

G. Lample, N. Zeghidour, N., Usunier, A., Bordes, L., Denoyer, & M.-A. Ranzato (2017),
Fader Networks Manipulating Images by Sliding Attributes,
In Computer Vision and Pattern Recognition, 2017.

Architectures
Fader network



G. Lample, N. Zeghidour, N., Usunier, A., Bordes, L., Denoyer, & M.-A. Ranzato (2017),
Fader Networks Manipulating Images by Sliding Attributes,
In Computer Vision and Pattern Recognition, 2017.

Architectures
Fader network

Encoder-decoder objective, with variables original input x and attribute y

$$\mathcal{L}_{\mathrm{AE}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) = \frac{1}{m} \sum_{(x,y)\in\mathcal{D}} \left\| D_{\theta_{\mathrm{dec}}}\left(E_{\theta_{\mathrm{enc}}}(x), y\right) - x \right\|_2^2$$

Idealistically, the decoder D(E(x), y) should be able to manipulate the attribute y while preserving other information of x

Without additional constraints, the attribute y has no effect on the decoder D(E(x), y)

Indeed, z = E(x) already contains the attribute information y

- One must learn a representation z invariant to y

Architectures
Fader network

To do so, one adds a discriminator module

Discriminator objective

$$\mathcal{L}_{\text{dis}}(\theta_{\text{dis}}|\theta_{\text{enc}}) = -\frac{1}{m}\sum_{(x,y)\in\mathcal{D}} \log P_{\theta_{\text{dis}}}\left(y\middle|E_{\theta_{\text{enc}}}(x)\right)$$

The discriminator tries to correctly predict the attribute **y** from the latent code **z** = E(**x**)

Given the encoder parameters, the discriminator parameters are updated to optimize the discriminator loss

Architectures
Fader network

The original auto-encoder is then rewritten with an additional adversarial loss

Adversarial objective

$$\mathcal{L}(\theta_{\text{enc}}, \theta_{\text{dec}} | \theta_{\text{dis}}) = \frac{1}{m} \sum_{(x,y) \in \mathcal{D}} \left\| D_{\theta_{\text{dec}}}\left(E_{\theta_{\text{enc}}}(x), y\right) - x \right\|_2^2 - \lambda_E \log P_{\theta_{\text{dis}}}(1 - y | E_{\theta_{\text{enc}}}(x))$$

The adversarial loss is composed of the original **reconstruction loss** (left term) and a **regularization loss** (right term)

The **regularization loss** is defined so that the discriminator is not able to predict the attribute **y** from the latent representation **z** = E(**x**).

In the original paper, **y** is a binary number (0 or 1) for each attribute, then 1 – **y** denotes a systematic wrong prediction

Ultimately, the latent representation z = E(x) does not contain information about y, but x can be reconstructed from z and y

Application to sound synthesis

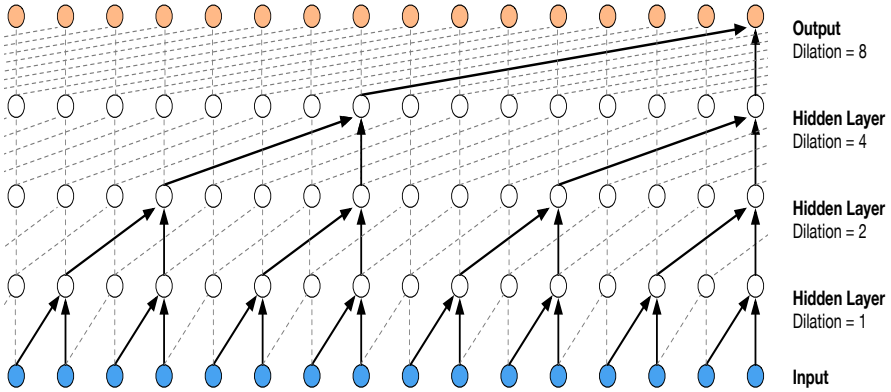
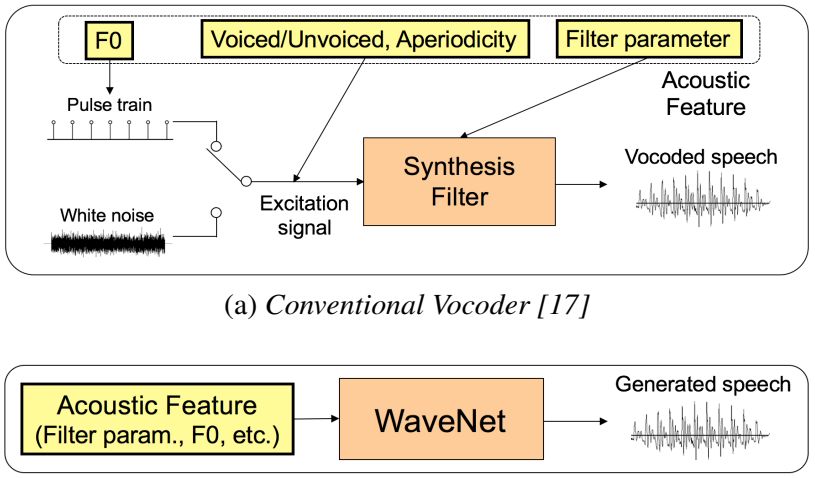


Figure 3: Visualization of a stack of *dilated* causal convolutional layers.



(a) *Conventional Vocoder [17]*

(b) *Proposed*

Figure 1: *Difference in waveform generation between conventional vocoder and proposed method [16]*

- WaveNet *(van den Oord et al., 2017)*
- Sound synthesis from raw waveform with auto-regressive generative model (1-D dilated convolutions 1-D)

- Conditional WaveNet *(Tamamori et al., 2017)*
- Conditioning a WaveNet on desired speech parameters (e.g., F0, intensity, etc...)

Application to sound synthesis

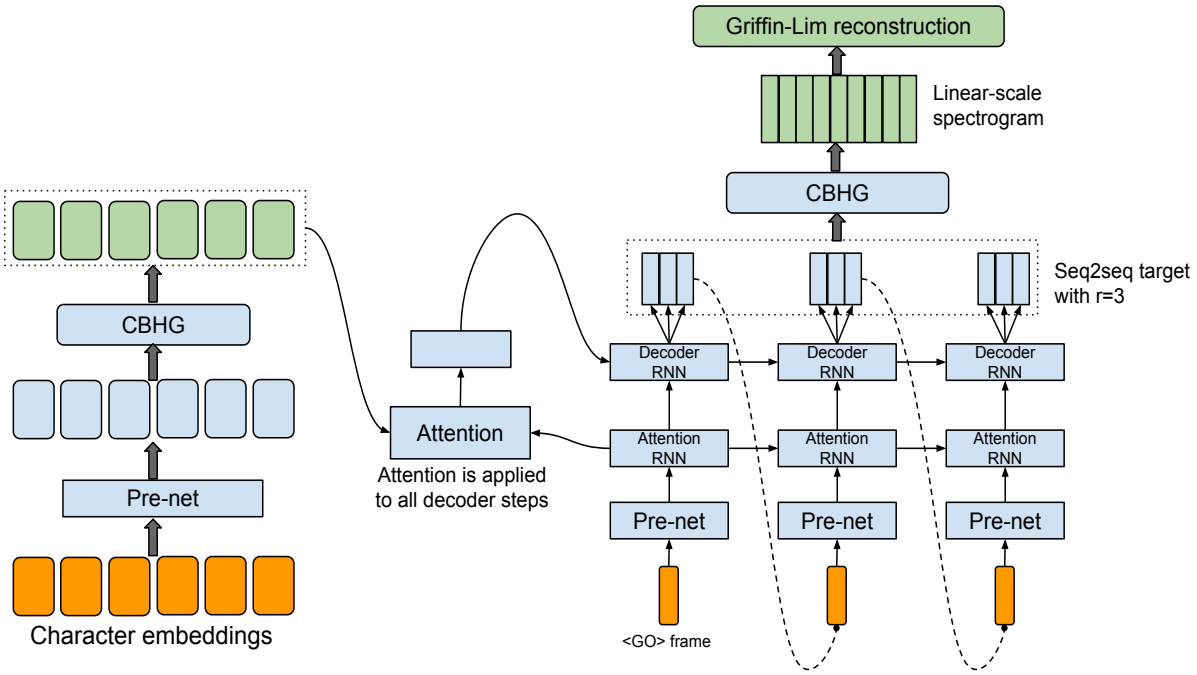


Figure 1: *Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.*

- Tacotron is a S2S network between a text sequence and an audio signal sequence (neuronal TTS)
- The decoder is an auto-regressive model conditioned on the encoding of the input text sequence

Application to sound synthesis

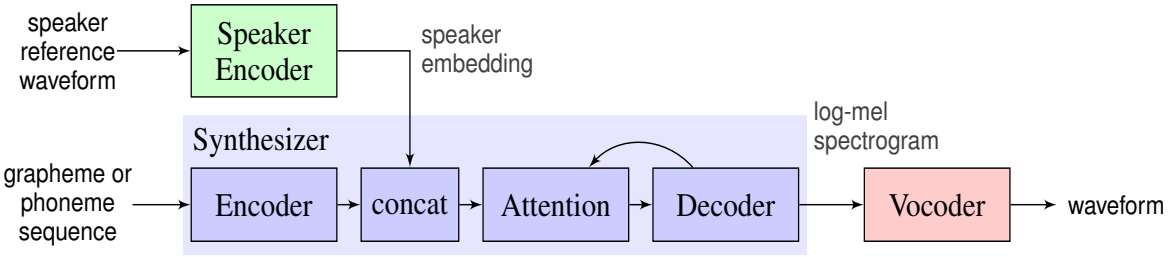


Figure 1: Model overview. Each of the three components are trained independently.

- Besides, this decoder can also be conditioned on other variables, such the **speaker identity**!

Y. Jia et al. (2018) Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis, proc. of NeurIPS (NIPS).
https://google.github.io/tacotron/publications/speaker_adaptation/

C8. Auto-encoders & generative adversarial networks

Advanced Machine Learning (MLA)

M2 Engineering of Intelligent Systems

& Advanced Systems and Robotics

2022-2023